BLZPACK: Description and User's Guide

Osni A. Marques

CERFACS Report TR/PA/95/30

BLZPACK: Description and User's Guide

Osni A. Marques[†]

October 1995 (revised June 1997)

Abstract

This report describes BLZPACK (an acronym for Block Lanczos Package), an implementation of the block Lanczos algorithm intended for the solution of eigenproblems involving real, sparse, symmetric matrices. The package works in an interactive way, so the matrices of the target problem are not passed as arguments for the interface subprogram. This means that each time an algebraic operation with the matrices of the eigenvalue problem has to be performed, the control is returned to the user (reverse communication strategy). The user is therefore free to exploit the characteristics of a particular application. The fundamentals of the technique implemented are first reviewed. Next, some practical details are outlined, such as the monitoring of the orthogonality of the Lanczos basis, the spectral transformation, the automatic spectrum slicing and the computational interval, and the vectors used in case of restart. Then, a set of applications and conclusions are presented. Finally, a user's guide is given in the Appendix.

[†]CERFACS, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, 42 av. G. Coriolis, 31057 Toulouse Cedex, France, e-mail: marques@cerfacs.fr

1 Introduction

The solutions of the equation

$$Ax = \lambda Bx \tag{1}$$

where A and B are $n \times n$ matrices, x is a non null vector and λ is a scalar, has long been an important computation. We recall that the previous relation defines a *generalized eigenproblem*. When B is equal to the identity matrix a *standard eigenproblem* is obtained. Usually, the n possible pairs (λ, x), *eigenvalues* λ and *eigenvectors* x, that satisfy (1) are associated with fundamental characteristics of differential and integral operators describing a physical phenomenon. In some applications from chemistry, for instance, they are related with basic configurations of molecules (hinge-bending motions); in structural engineering, dynamic properties of a given model (natural vibration frequencies and mode shapes); and in nuclear power plants, neutron fluxes (the behaviour is super-critical for a dominant eigenvalue greater than one). Depending on the complexity, the level of the discretization of a continuous problem or the precision required for the results, A and B can reach dimensions of tens of thousands. In practical analyses, only a subset of the n *eigenpairs* (λ, x) is considered relevant, either in the extremities of the spectrum or in an interval $[\xi_1, \xi_2]$. Although only a few eigenpairs may be wanted, their evaluation is usually a time consuming task. The development of new eigensolvers, or the improvement of existing ones, has been thus the subject of continuous research [14, 28, 38, 40].

Nowadays, Krylov subspaces based methods such as Lanczos [23] and Arnoldi [2] algorithms are widely used for treating eigenproblems associated with large sparse matrices. They can be shown to perform better than vector iteration (inverse or direct), transformation methods (Jacobi, Householder or Givens) or determinant search in many practical cases. See [15] and [33] for an overview of those techniques. The Krylov subspace associated with a matrix A and a starting vector q_1 of unitary length is defined as

$$\mathcal{K}(A, q_1, j) = span(q_1, Aq_1, ...A^{j-1}q_1). \tag{2}$$

It turns out that the j-th vector of the sequence converges toward the eigenvector associated with the dominant eigenvalue [15, 33]. The goal of the Lanczos algorithm is the generation of a basis for a Krylov subspace. The projection of the original problem into the basis leads to a smaller problem, involving a tridiagonal matrix (which is symmetric if A is). Eigensolutions are then recovered through a Rayleigh-Ritz procedure [33]. Conversely, the projection computed by Arnoldi's method corresponds to a Hessenberg matrix [15].

A set of implementations of the Lanczos/Arnoldi algorithm have been proposed for real symmetric matrices [6, 7, 13, 19, 27, 32, 39]. Four packages, for instance, are of public domain and available at Netlib [4]: ARPACK, developed by Sorensen and Lehoucq [24, 40], LANCZOS, developed by Cullum and Willoughby [8], LANZ, developed by Jones and Patrick [20, 21, 22], and LASO, developed by Scott [34]. One of the most robust implementations was performed by Grimes, Lewis and Simon [16, 17, 18] which is incorporated in the MSC/NASTRAN structural analysis code, for example, and also in some scientific libraries. The main feature of the implementation is a combination of a block Lanczos strategy with a dynamic shift-invert scheme. Associated with the block strategy there is a Krylov subspace built from a full rank $n \times p$ matrix $Q_1 = [\ q_1^{(1)}\ q_2^{(1)}\ ...\ q_p^{(1)}\ ]$, $Q_1^T Q_1 = I$, $1 < p \ll n$, where p is the *block size* [15, 33]:

$$\mathcal{K}(A, Q_1, j) = span(Q_1, AQ_1, ...A^{j-1}Q_1). \tag{3}$$

The approach by blocks allows for better convergence properties when there are multiple eigenvalues and also a better data management on some computer architectures. A dynamic shift-invert (translation of origin and inversion of the eigenvalue spectrum) is useful when many eigenvalues are sought or the eigenvalue distribution is clustered.

The code described in this work corresponds to an implementation of the block Lanczos algorithm intended for the solution of the problem $Ax = \lambda Bx$, with A and B symmetric. It is assumed that there exists a linear combination of A and B which is positive definite, to assure that all eigenvalues are real (see [33, Chapter 15] for details). The development of the code was originally motivated by the study of free vibration problems in structural engineering (mainly large frame and shell structures). Upgrades were performed afterwards aiming at applications from different fields. This was an incentive to describe and document the code appropriately. The present version, BLZPACK, aims to be a user-friendly tool for distinct problems involving real symmetric matrices. The package is tailored to a class of applications for which at least one "inversion" of the operator $A_\sigma = A - \sigma B$ is feasible, where σ is a real scalar. With such an inverted operator the eigenvectors are preserved while the eigenvalues are remapped. Eigenvalues lying in a range of interest can be then set apart from the remaining eigenvalues, leading to better converge rates for the wanted solutions. In practical cases, the inversion is replaced by a factorization of A_σ for the solution of systems of linear equations. For some applications, the factorization may dominate the computational costs, but it is often the only way to deal efficiently with the companion eigenvalue distributions. The most important features of BLZPACK are the following:

- The matrices A and B are not passed as parameters, all algebraic operations with them should be performed by the user outside the code. In other words, each time a computation involving either A or B has to be performed, the control is returned to the user. Such a computation can be then specialized for particular applications.
- An automatic spectrum slicing scheme may be activated by the user for problems of the type $Ax = \lambda Bx$. This option may require factorizations of matrices $A - \sigma B$ for a set of real scalars σ, but provides for an improvement in the convergence rate and reliable checks in the computed solutions.
- By setting an appropriate flag, the code can be applied to either $Ax = \lambda x$ or $Ax = \lambda Bx$. In the former case, the code just skips over steps that are normally required for the generalized eigenproblem, thus saving in operations.

In the next section, we briefly describe the governing ideas of the Lanczos method and the implementation performed. We have opted for not giving exhaustive demonstrations. The most important reason for it is that good references exist, so we prefer to "forward" the reader to them when the occasion appears. On the other hand, we list some implementation details to help the reader to understand how the code works. In the applications section, we propose three examples to illustrate the usage of the package: a benchmark coming from a finite element structural analysis, with multiple and clustered solutions; a SVD estimation for a rectangular matrix related with an economics analysis model; and a problem related with the study of motions of a protein. We also include a set of cases showing the effects of the block size and the spectrum slicing strategy. Finally, we give a user's guide for the package, with two driver models.

2 The Technique

In this section we summarize the fundamentals of the block Lanczos algorithm. We begin with the formulation for the generalized eigenproblem, comment on a positive semidefinite matrix B, and then particularize the formulation for the standard problem. The objective is to build an appropriate basis for the Krylov subspace defined in (2). We recall that the Gram-Schmidt orthonormalization process could be applied to all vectors of that subspace, in the natural order $q_1, Aq_1, A^2q_1, ...$, so as to construct a basis for it [33]. Nevertheless, one can show that for the orthonormalization of the i-th Krylov vector, it suffices to take into account only the two previous orthonormalized vectors. Moreover, the basis can be built vector by vector, without generating the Krylov subspace. That is exactly what the Lanczos method allows us to compute, the process is similar for the subspace defined in (3) [15, 30, 33, 35].

Considering a real scalar σ different from an eigenvalue, we can rewrite the problem (1) as

$$BA_\sigma^{-1}Bx = (\frac{1}{\lambda - \sigma})Bx \tag{4}$$

where $A_\sigma = A - \sigma B$. Therefore, both the symmetry and the eigenvector are preserved, the eigenvalue being now given by $\frac{1}{\lambda-\sigma}$. It is usually assumed (and observed) that the Lanczos algorithm finds eigenpairs at both ends of the spectrum. The convergence pattern, however, is mainly determined by the eigenvalue distribution. With the shift of origin and inverse formulation indicated in (4), the eigenvalues around σ are set apart from the others, namely to the extremes of a modified spectrum, as depicted in Figure 3 (see [13], [31] and [36] for a discussion on inverted operators). Note that the problem (4) can be simplified to $A_\sigma^{-1}Bx = (\frac{1}{\lambda-\sigma})x$: the operator $A_\sigma^{-1}B$ is nonsymmetric but it is self-adjoint with respect to B and its eigenvalues are thus real. The block Lanczos basis generation strategy for this latter problem is summarized in Table 1. Needless to say, A_σ^{-1} is not explicitly evaluated due to numerical stability problems and storage requirements, but that notation make us remember of the operator that would be used for building the Krylov subspace. On the other hand, the operator A_σ can be factorized as LDL^T, where L is a lower unit triangular matrix and D is a direct sum of 1×1 and 2×2 pivot blocks, allowing for solutions of systems of linear equations for the basis generation.

As can be seen in Table 1, the Lanczos basis is constructed from a rank p starting block R_0 (possibly defined with random entries) which leads to Q_1. Basically, at each step, a simultaneous inverse iteration on the product BQ_j yields the "residual" R_j, which is orthogonalized against the two previous blocks, Q_{j-1} and Q_j. The factorization of the residual, through a modified Gram-Schmidt process, results in the next set of p B-orthogonal vectors, Q_{j+1}. Thus, Q_j and R_j are $n \times p$ matrices, A_j is $p \times p$ and B_j is $p \times p$ upper triangular, defined as

$$\mathsf{A}_j = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & ... & \alpha_{1,p} \\ \alpha_{2,1} & \alpha_{2,2} & ... & \alpha_{2,p} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \alpha_{p,1} & \alpha_{p,2} & ... & \alpha_{p,p} \end{bmatrix}, \qquad \mathsf{B}_j = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & ... & \beta_{1,p} \\ 0 & \beta_{2,2} & ... & \beta_{2,p} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 0 & 0 & ... & \beta_{p,p} \end{bmatrix}, \tag{5}$$

and in a finite precision arithmetic one can write (since Q_{j-1} and Q_j are orthogonal)

$$R_j = Q_{j+1}\mathsf{B}_{j+1} = A_\sigma^{-1}BQ_j - Q_j\mathsf{A}_j - Q_{j-1}\mathsf{B}_j^T. \tag{6}$$

Table 1: Block Lanczos Algorithm

set $Q_0 = 0$
set $R_0 \neq 0$ and factorize $R_0 = Q_1 \mathsf{B}_1$, $Q_1^T B Q_1 = I$
define σ and $A_\sigma = A - \sigma B$

for j=1,2,...
a) compute $R_j = A_\sigma^{-1} B Q_j$
b) $R_j \leftarrow R_j - Q_{j-1} \mathsf{B}_j^T$
c) $\mathsf{A}_j \leftarrow Q_j^T B R_j$
d) $R_j \leftarrow R_j - Q_j \mathsf{A}_j$
e) factorize $R_j : R_j = Q_{j+1} \mathsf{B}_{j+1}$, $Q_{j+1}^T B Q_{j+1} = I$
end for

After j steps, the blocks of vectors generated can be arranged as

$$\mathcal{Q}_j = \left[\begin{array}{cccc} Q_1 & Q_2 & \ldots & Q_j \end{array} \right] \tag{7}$$

satisfying

$$\mathcal{Q}_j^T B \mathcal{Q}_j = I \tag{8}$$

where I is an identity matrix with $j \times p$ rows, and

$$\mathcal{Q}_j^T B A_\sigma^{-1} B \mathcal{Q}_j = T_j, \tag{9}$$

where

$$T_j = \left[\begin{array}{ccccc} \mathsf{A}_1 & \mathsf{B}_2^T & & & \\ \mathsf{B}_2 & \mathsf{A}_2 & \mathsf{B}_3^T & & \\ & \mathsf{B}_3 & \mathsf{A}_3 & \ddots & \\ & & \ddots & \ddots & \mathsf{B}_j^T \\ & & & \mathsf{B}_j & \mathsf{A}_j \end{array} \right]. \tag{10}$$

Therefore, the projection of the eigenproblem (4) into the basis defined by relation (7) is the symmetric block tridiagonal matrix T_j, with an approximate solution

$$(\hat{\lambda}_k, \hat{x}_k) \tag{11}$$

for (1) or (4) given by the *Ritz value*

$$\hat{\lambda}_k = \sigma + \frac{1}{\theta_k}, \tag{12}$$

and by the *Ritz vector*

$$\hat{x}_k = \mathcal{Q}_j s_k, \tag{13}$$

where (θ_k, s_k) is the solution of the reduced problem

$$T_j s_k = \theta_k s_k. \tag{14}$$

The $j \times p$ solutions of the problem (14) can be obtained by first reducing T_j to tridiagonal form, for example, and $\hat{x}_k^T B \hat{x}_k = 1$ providing $s_k^T s_k = 1$. Certainly, for matrices of same dimension the computational cost for solving (14) increases with the block size p. In many cases, some extreme eigenvalues of T_j lead to good approximations of some extreme eigenvalues of problem (4) with $j \ll n$. Moreover, the norm of the residual vector

$$BA_\sigma^{-1} B\hat{x}_k - (\frac{1}{\hat{\lambda}_k - \sigma}) B\hat{x}_k$$

can be estimated *a priori*, through

$$\|A_\sigma^{-1} B\hat{x}_k - (\frac{1}{\hat{\lambda}_k - \sigma})\hat{x}_k\| = \|Q_j(T_j - \theta_k s_k) + R_j E_j s_k\| = \|R_j E_j s_k\| = \|\mathsf{B}_{j+1} s_j^{(k)}\| = \beta_j^{(k)}, \quad (15)$$

where $\|\,.\,\|$ is the Euclidian norm with respect to B, E_j is a block matrix with the j-th block equal to the identity matrix and the others zero, and $s_j^{(k)}$ stores the bottom p elements of s_k. Therefore, the monitoring of the norm $\beta_j^{(k)}$ allows the counting of the converged solutions, by comparing with a specified tolerance, and the finalization of a given run.

The Modified Gram-Schmidt Factorization. The Gram-Schmidt factorization of R_j as $Q_{j+1}\mathsf{B}_{j+1}$, in step (e), Table 1, consists basically in the B-normalization of a column of R_j and the purging of this column from the remaining ones. The process starts with the first column and so forth. If the product BR_j is available, its columns can be modified in a similar way. At the end, R_j and BR_j are overwritten by Q_{j+1} and BQ_{j+1}, the diagonal of B_{j+1} stores the normalizing factors and its upper triangle the purging factors as defined in (5). However, one sweep on the p columns of R_j may be not enough to produce a $\|Q_{j+1}^T BQ_{j+1} - I\|$ which is negligible. Therefore, the factorization should be performed in an iterative way, repeated up to $p \times 2$ times if required. After the factorization, an extra orthogonalization is usually performed between Q_j and Q_{j+1} to guarantee the local level of orthogonality. If the B-norms of the columns of R_j are tiny, and consequently $\|\mathsf{B}_{j+1}\|$ drops to zero, an invariant subspace has been computed. It means that all solutions of the reduced problem lead to solutions of the original problem. Such a situation is clearly reached when $j \times p = n$. The residual block may also result rank deficient if σ is applied too close to an eigenvalue, the starting vectors are linearly dependent or $j \times p > n$ (this case is likely to happen for small problems). In addition, if the matrix B is diagonal with k non null entries, $k < n$, that would also be the maximum size for the basis of Lanczos vectors. In order to verify the conditioning of R_j one can compute the singular value decomposition of B_{j+1}. This calculation is rather inexpensive because p is usually very small when compared to n. See [5, 10, 18] for details.

A Positive Semidefinite B. A positive semidefinite matrix B in the generalized eigenproblem implies in the existence of eigenvalues equal to $-\infty$ or $+\infty$. However, the formulation presented heretofore still holds, providing the Lanczos vectors and eigenvectors are kept in the proper space, in other words, purged from the unwanted components in the null space of B [31]. In order to perform that, the matrix B is first applied to R_0, i.e., R_0 is replaced by BR_0 in Table 1. Then, one step of inverse iteration is applied to each converged Ritz vector, which is equivalent to update $\hat{x}_k$ in relation (13) with

$$\hat{x}_k \leftarrow \hat{x}_k + \frac{1}{\theta_k} Q_{j+1} \mathsf{B}_{j+1} s_j^{(k)}. \quad (16)$$

The Standard Problem. In this case, the matrix B is simply replaced by the identity matrix and the generation of the basis requires less operations. In addition, depending on the application, no spectral transformation is needed and step (a) in Table 1 can be rewritten as $R_j = A_\sigma Q_j$. A Ritz value would be then obtained as $\hat{\lambda}_k = \sigma + \theta_k$, the corresponding Ritz vector by relation (13). That translation, however, is not likely to improve the convergence around σ as the combination translation-inversion is. Therefore, we can just set σ to zero. See [33, page 63] for a discussion on shifts of origin.

3 Implementation Details

This section outlines some aspects of the implementation of BLZPACK. The ideas given in [13], [16, 17, 18], [30], [20, 21, 22] and [34, 37] complement the subjects discussed here. We start with the monitoring of the basis orthogonality and then comment on the data management during the basis generation process, the spectral transformation, the spectrum slicing strategy and the computational interval, and the vectors used in case of a restart.

3.1 Monitoring the orthogonality of the basis

A loss of orthogonality among the vectors of the basis $\mathcal{Q}_j$ is generally observed after some steps. It is caused by roundoff errors, which could be represented by an additional term, say F_j, in relation (6). On the other hand, Paige [33, page 264] showed that the departure from orthogonality is also related to the convergence of a pair $(\hat{\lambda}_k, \hat{x}_k)$ and, therefore, with the eigenvalue distribution of the associated problem. Once orthogonality is lost, property (8) is no longer satisfied, and redundant copies of eigenpairs emerge. As an immediate option to avoid this, one could apply a full reorthogonalization, i.e., orthogonalize R_j against all $j-1$ previous blocks. However, such a scheme would strongly increase the number of operations at each step. On the other hand, some preventive measures based on potentially dangerous vectors can be used to keep the basis orthogonality within a certain level, namely:

Selective Orthogonalization. The selective orthogonalization aims at keeping the columns of R_j orthogonal to all converged $\hat{x}_k$. Therefore, whenever the error $\beta_j^{(k)}$ satisfies a specified tolerance, the corresponding Ritz vector $\hat{x}_k$ is calculated and stored. The tolerance for $\beta_j^{(k)}$ is usually set to $\sqrt{\epsilon}\|A\|$, where $\|A\|$ may be estimated through T_j and ϵ is the relative machine precision. In the following steps, the residual block R_j is orthogonalized against $\hat{x}_k$ whenever indicated by the norm $\tau_{j+1}^{(k)} = \| \hat{x}_k^T B Q_{j+1} \|$. This norm can be estimated at almost no cost, without explicitly using $\hat{x}_k$. See [18, 34] for details.

Partial Reorthogonalization. The objective of the partial reorthogonalization is to keep R_j orthogonal to all Q_i, $i < j$. The level of orthogonality among the blocks of vectors is then measured at each step through the norm $\eta_{i,j+1} = \| Q_i^T B Q_{j+1} \|$, and whenever it is greater than a given threshold, Q_{j+1} is orthogonalized against the corresponding Q_i. The procedure can be seen as a repetition of steps (c) and (d) in Table 1, with Q_i instead of R_j. The norm $\eta_{i,j+1}$ can be estimated at almost no cost, without explicitly computing matrix products with Q_i or Q_j. See [18, 37] for details.

A reasonable strategy is to reorthogonalize whenever $\tau_{j+1}^{(k)}$ or $\eta_{i,j+1}$ are greater than $\sqrt{\epsilon}np$. Nevertheless, it is likely that the effectiveness of the selective orthogonalization or the partial reorthogonalization depends on the application and such strategies are sometimes used together [16, 17, 18, 30]. Actually, in the present implementation, we adopt the strategy proposed by Grimes *et al.* [18] and compute eigenvectors only at the end of a given run. A selective orthogonalization will be then performed only if a restart is required. Still, a modified partial reorthogonalization strategy is used, in the sense that whenever $\eta_{i,j+1}$ reaches the threshold, all the previous Q_j are taken into account [30]. Therefore, instead of partial reorthogonalizations at possibly close steps, a full reorthogonalization is performed at some steps. Finally, for both orthogonalization schemes, both Q_j and Q_{j+1} are orthogonalized, since they are needed in the computation of Q_{j+2}. Such a strategy assures an adequate level of orthogonality for some subsequent steps. It should be noted that the number of extra operations introduced by the selective orthogonalization vary with the number of converged eigenpairs. Conversely, the extra operations introduced by the partial reorthogonalization vary with the number of Lanczos steps. In addition, for bases of same size, the operations for orthogonalizing Q_j and Q_{j+1} are proportional to p. Another important aspect is the way the Lanczos vectors are stored. As a result, for long Lanczos runs the orthogonality control may correspond to an important share of the computational costs.

3.2 Data Management

BLZPACK requires four two-dimensional arrays of dimensions $n \times p$ to generate the basis of Lanczos vectors for a generalized eigenproblem. Instead of swapping entries between those four arrays, the package uses four pointers and an array of length $n \times p \times 4$. Then, the pointers exchange their values at each step, indicating the correct positioning of those four entities inside the one dimensional array. Let us consider the scheme depicted in Figure 1. At "step 0", the start-up, the first pointer points to R_0 and the third pointer to BR_0. Those matrices are transformed into Q_1 and BQ_1, respectively, after an orthonormalization of their columns by means of a Gram-Schmidt process. For the next (first) step, the first pointer receives the value of the second, the second pointer the value of the third, the third pointer the value of the fourth, and the fourth pointer the previous value of the first. The notations Q_0 and BQ_0 were introduced only to clarify the explanation: their virtual positions are overwritten by R_1 (computed from BQ_1, which is addressed to by the second pointer) and BR_1. Then, these matrices lead to Q_2 and BQ_2, respectively. The Lanczos process and exchanges of pointers continue so that when the block Q_{j+1} is computed, Q_j can be moved into the space reserved for the basis (Q_{j+1} may be required in the next step for the orthogonality control scheme). This is one of the swaps of $n \times p$ entries performed by the package at each step. The other is related with the reverse communication strategy, to supply data to the user and obtain results back after the computations involving A and B. Note that the overhead at this point has been introduced for the sake of interface simplicity. The pointer strategy for the standard problem is illustrated in Figure 2: at "step 0" the first pointer points to R_0, which leads to Q_1. In the next step, the third pointer receives the value of the second, the second pointer the value of the first, and the first pointer the previous value of the third. In this case, however, the basis is built using only 3 arrays of dimensions $n \times p$. The fourth pointer is kept as well, which always receives the same entry of the second pointer. With four pointers, the same computational module can be used to perform a Lanczos step for both types of eigenproblem.

3.3 Spectral Transformation

With a shift-invert strategy, difficult eigenvalue distributions can be remapped in a more favorable way. In other words, the eigenvalues of the problem involving the block tridiagonal matrix T_j will correspond to $\theta = \frac{1}{\lambda - \sigma}$. The Figure 3 illustrates the new situation: solutions close to the translation of origin σ are better separated in terms of θ, solutions far away from σ are grouped around 0. Let us assume that there is an eigenvalue λ close to σ with the corresponding approximation computed by the Lanczos algorithm given by $\hat{\lambda}$, i.e., $\hat{\lambda} = \sigma + \frac{1}{\theta}$. It turns out that even a moderate value of the residual error norm (15) suffices to guarantee that $\hat{\lambda}$ is a good approximation for λ, in contrast with the solutions far from σ. One can show for example that

$$|\lambda - \hat{\lambda}| \leq \frac{\beta_j}{\theta^2},$$

where the indexes of β_j and θ where dropped for the sake of generality. That bound can be further improved for well-separated eigenvalues. See [13], [18] and [31] for details.

Given an interval $[\xi_L, \xi_R]$ containing σ, there are two corresponding bounds in θ, θ_L and θ_R, as can be seen in Figure 3. An eigenvalue lying in that interval must have a counterpart θ_k satisfying either $\theta_k \leq \theta_L$ or $\theta_k \geq \theta_R$. A similar condition can be then used to accept and discard eigenvalue approximations when a spectrum slicing scheme is applied. The spectral transformation offers an additional feature when implemented in terms of the factorization $A_\sigma = LDL^T$, for the solution of systems of linear equations. One can show that the inertias of A_σ and D are the same [15]. We recall that the inertia of a matrix is given by a triple of nonnegative integers: the number of eigenvalues less, equal, and greater than zero [15, 33]. Therefore, the number of eigenvalues in $[\xi_L, \xi_R]$ is simply given by the difference $nd_R - nd_L$, where nd_R and nd_L are the number of negative 1×1 plus 2×2 pivots in D [11], for $\sigma = \xi_R$ and $\sigma = \xi_L$, respectively. Such a test, also referred to as Sturm sequence check [3], allows as well the location in the spectrum of any pair $(\hat{\lambda}_k, \hat{x}_k)$.

3.4 Spectrum Slicing and Computational Interval

A spectrum slicing strategy is useful when many eigensolutions are required, the eigenvalue distribution is clustered (such that the converge is slow), or the continuation of a given run is expensive when compared with the generation and factorization of a new operator A_σ. Furthermore, long Lanczos runs tend to require increasing reorthogonalization efforts (both selective orthogonalization and partial reorthogonalization). In such cases, it would be preferable to restart with a new origin, as an attempt to save Lanczos steps and modify the convergence rate. The eigenproblem would be then solved by pieces, based on the spectrum that the Lanczos algorithm approximates as the basis sizes increase. However, factorizations $A_\sigma = LDL^T$ for a set of scalars σ might be required and a compromise between the factorization costs and convergence rates should be established. Since there is no definitive criterion to decide when a σ should be abandoned and a new factorization computed, a decision based on heuristics or computational costs is usually employed. Grimes *et al.* [18], for instance, model the cost for continuing the Lanczos recurrence beyond the current step and try to locate a point in an individual run for which the average cost per eigenvalue is minimized. In the BLZPACK implementation the criterion is based on CPU times and convergence rates.

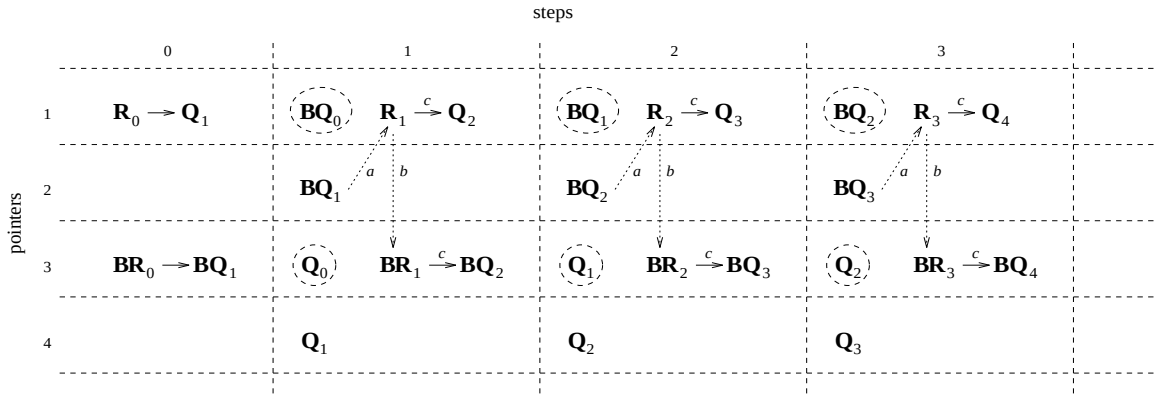

Figure 1: Data management for $Ax = \lambda Bx$.

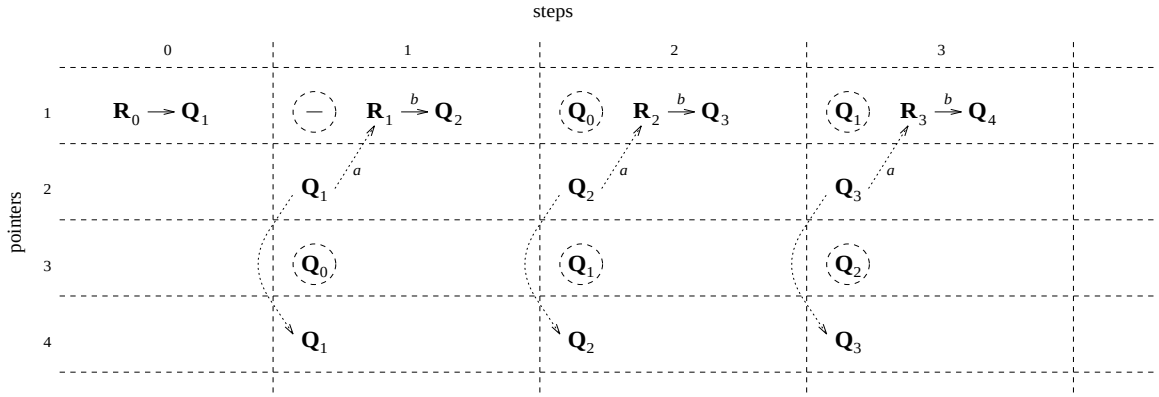

Figure 2: Data management for $Ax = \lambda x$.

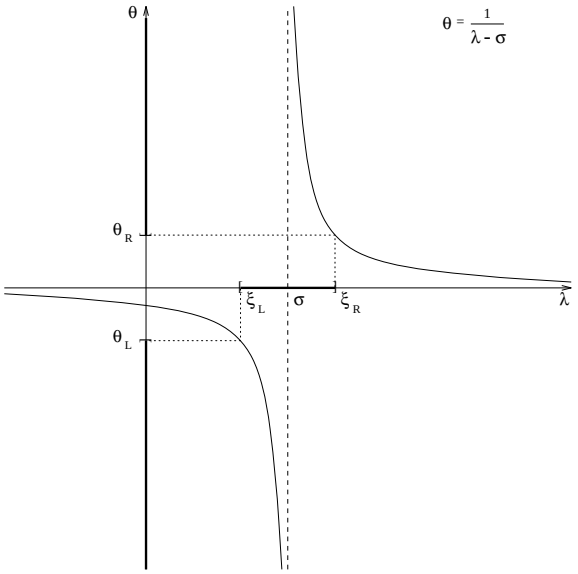

Figure 3: The transformation of the spectrum.

At each step, the combination of three factors is taken into account to verify how well the algorithm is doing and to decide between continuing or restarting:

1. The $p \times 2$ smallest residuals $\beta_j^{(k)}$ that did not satisfy the tolerance for convergence are examined, compared with those of the previous step and extrapolated to check if they tend to decrease in the next step.
2. The CPU time spent with the basis generation is compared with the CPU time spent with the basis orthogonality control.
3. The CPU time spent with the basis generation is compared with the CPU time spent with the factorization $A_\sigma = LDL^T$.

These factors are then weighed and combined. If the result is greater than a certain threshold, a new factorization is assumed to be feasible. There is, however, a minimum number of steps to be performed before σ can be modified. The minimum dimension for the Lanczos basis is set to 40 if $p = 1$, 50 if $p = 2$, 60 if $3 \leq p \leq 6$, and 80 otherwise. Such a constraint aims at gathering information from the eigenvalue spectrum to be used in the definition of the next σ. Another condition for restarting happens when σ is taken too close to an eigenvalue and such a quasi-singularity is not detected during the factorization of A_σ. We assume that the distribution of the eigenvalues of the reduced problem are likely to indicate that and override the aforementioned conditions in order to avoid numerical difficulties. Conversely, one of the objectives of the shift selection implemented is exactly not to take a value too close to an eigenvalue. The rule is violated only when clusters of eigenvalues are detected.

The value of a new σ is defined considering the approximate eigenvalue distributions computed with the previous σ's. The adopted strategy is similar to the one proposed by Grimes *et al.* [18]. Usually, around σ_i there is a set of values, $\hat{\lambda}$, which are accepted as approximate eigenvalues. There is also another set, contiguous to the prior, that does not satisfy the convergence criterion but gives an indication of the eigenvalue distribution. Let us call μ the values in this set. Assuming that k eigenvalues have been found to the *right* of σ_i, the objective is to define σ_{i+1} in such a way that k eigenvalues will remain to be computed to its *left*. Thus, σ_{i+1} bisects μ_k and μ_{k+1}. If required, σ_{i+1} can be placed to the left of σ_i using a similar approach. In some cases, the information provided by the values μ is not enough to define σ_{i+1}. Then, σ_{i+1} will be specified using δ_{max}, the "radius of convergence". Let us assume that $\hat{\lambda}_{L,i}$ and $\hat{\lambda}_{R,i}$, are the farthest approximate eigenvalues computed with σ_i, to its left and to its right, respectively. For each σ_i we define $\delta_i = max(|\sigma_i - \hat{\lambda}_{L,i}|, |\sigma_i - \hat{\lambda}_{R,i}|)$, and $\delta_{max} = max(\delta_i)$, $\sigma_{i+1} = \sigma_i \pm \delta_{max} \times 2$, where the sign depends on the side of σ_i the new origin will be placed. Each translation σ_i defines two subintervals. In general, the spectrum slicing strategy sets for a Lanczos run trying to close a subinterval, by determining all eigenvalues lying in it. If the run succeeds, another subinterval is examined, otherwise the subinterval is either kept or split. The decision between keeping or splitting is based on the number of missing eigenvalues in the subinterval, which can be determined by the inertia information from A_σ, for $\sigma = \sigma_L$ and $\sigma = \sigma_R$, where σ_L and σ_R are the left (lower) and right (upper) limits of the subinterval. For a subinterval splitting, σ_{i+1} is defined as the average of the farthest eigenvalue approximations obtained to the right of σ_L and to the left of σ_R. The average can be either arithmetic or geometric, depending on the magnitude of the entries. If such values overlap each other, the limits of the subinterval are used instead.

The computation interval for BLZPACK may be specified as a pair $[\xi_1, \xi_2]$. Assuming that the number of required eigenpairs is given by m, we have three possibilities:

$\xi_1 > \xi_2$, the code runs from ξ_1 to ξ_2 (ξ_1 is the upper bound), seeking for the first m eigenpairs to the left of ξ_1 (Figure 4). The computation finishes if more than m eigenpairs exist in the interval or if an eigenvalue less than ξ_2 is found.

$\xi_1 < \xi_2$, the code runs from ξ_1 to ξ_2 (ξ_1 is the lower bound), seeking for the first m eigenpairs to the right of ξ_1 (Figure 5). The computation finishes if more than m eigenpairs exist in the interval or if an eigenvalue greater than ξ_2 is found.

$\xi_1 = \xi_2$, the code runs around ξ_1, seeking for the m eigenpairs closest to ξ_1 (Figure 6).

When $\xi_1 \neq \xi_2$ the goal is to move σ toward ξ_1 or ξ_2, depending on which is bigger as previously mentioned. When $\xi = \xi_1 = \xi_2$ the code starts with the subintervals $(-\infty, \xi]$ and $[\xi, +\infty)$ and the goal is to move σ around ξ. The Figures 4, 5 and 6 illustrate the procedure. We define a special variable INFO, to store information for each subinterval k as follows:

INFO(L,k): stores $\sigma_{L,k}$ (the lower limit of the subinterval), the number of eigenvalues less than $\sigma_{L,k}$ (from the inertia information), the farthest converged eigenvalue to the right of $\sigma_{L,k}$, and the new origin translation to be applied to the right of $\sigma_{L,k}$.

INFO(R,k): stores $\sigma_{R,k}$ (the upper limit of the subinterval), the number of eigenvalues less than $\sigma_{R,k}$ (from the inertia information), the farthest converged eigenvalue to the left of $\sigma_{R,k}$, and the new origin translation to be applied to the left of $\sigma_{R,k}$.

Around each σ in those figures there is a region indicated by brackets in which eigenvalues have been found (the set $\hat{\lambda}$). A new σ can be then used either to split a subinterval with missing solutions, for example σ_3, or to extend a subinterval in which all solutions were found, for example σ_4. When the upper (or lower) limit of a subinterval is moved too far, which means that there are much more eigenvalues to be computed to its left (or right) than the number required, the factorization is discarded and a new origin is chosen. A subinterval that has been closed is indicated by a bold line over the eigenvalue distribution. The dashed lines indicate how the contents of INFO are updated as the subintervals are opened and closed.

3.5 Vectors for restarting

The number of steps required for the convergence of the required solutions depends on the starting vectors and mainly on the eigenvalue distribution of the problem being solved. When the maximum number of steps allowed is reached, or a spectrum slicing is applied, without computing all required solutions, a restart is performed. Then, the code takes as initial vectors a linear combination of $p \times 2$ vectors corresponding to the smallest residuals $\beta_j^{(k)}$ that did not satisfy the converge criterion in the previous run. The Figure 7 shows values associated with the vectors considered: a) when a subinterval is split, b) when the origin is kept fixed, and c) when a standard problem is being solved. Again, brackets indicate regions where eigenvalues have been found. Therefore, the vectors considered are those corresponding to the values included in the set μ.

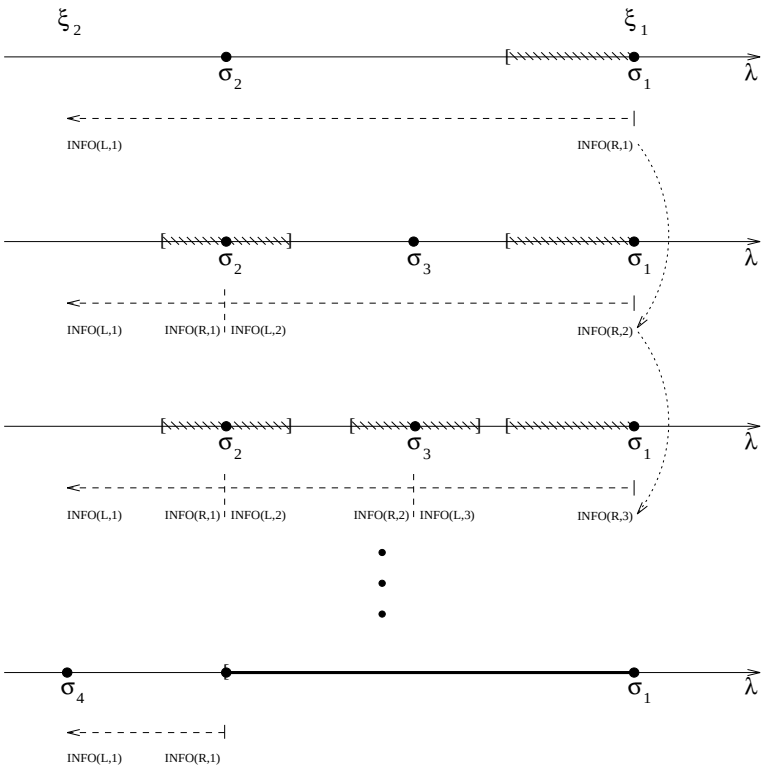

Figure 4: Computation interval $[\xi_1, \xi_2]$, run from the right to the left.

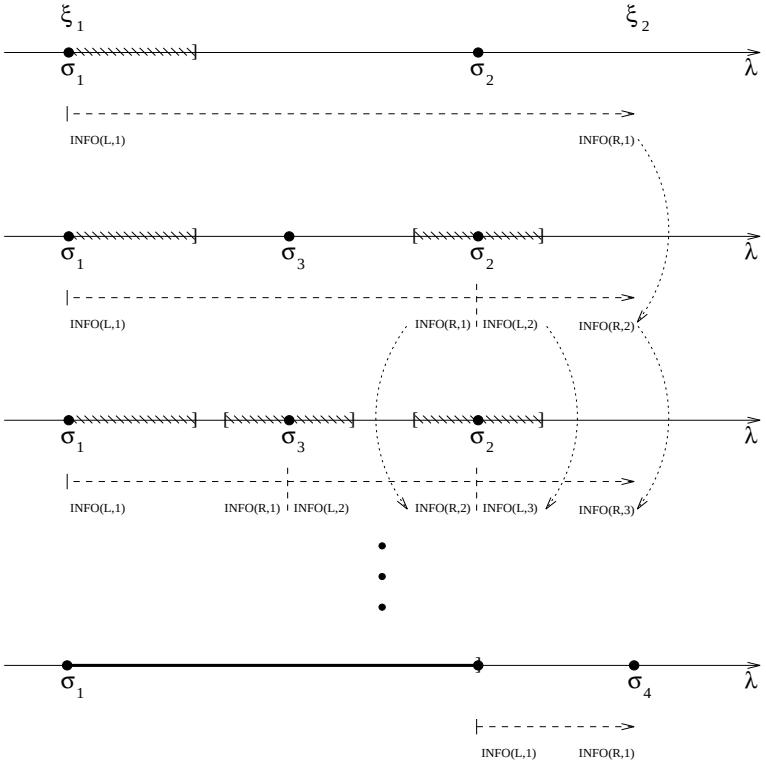

Figure 5: Computation interval $[\xi_1, \xi_2]$, run from the left to the right.

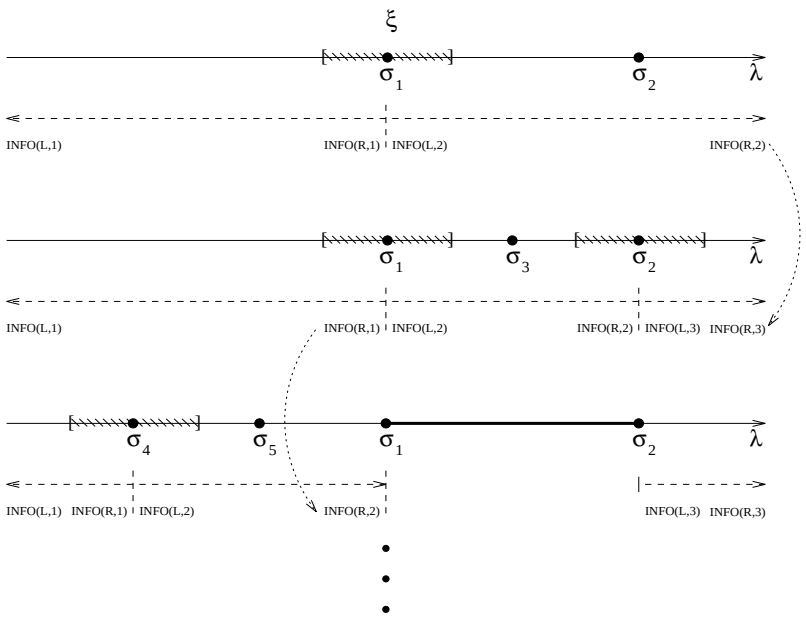

Figure 6: A reference value, run around ξ ($\xi_1 = \xi_2$).

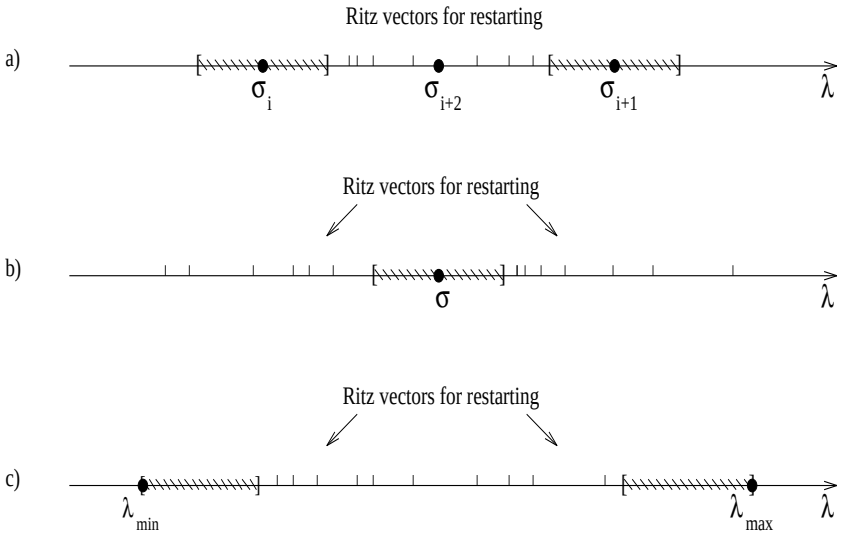

Figure 7: Vectors for restarting: a) splitting a subinterval, b) a fixed σ, c) $Ax = \lambda x$.

4 Applications

In this section, we give three examples to illustrate the usage of BLZPACK: a benchmark coming from a finite element structural analysis; a SVD estimation for a rectangular matrix coming from an economics analysis model; and a problem related with the investigation of motions of a protein. We also include a set of cases available in the Harwell-Boeing Sparse Matrix Collection [12], to show the effects of the block size and the spectrum slicing strategy. We use the notation of the previous sections: m denotes the number of required solutions, n the dimension of the matrices and p the block size for the Lanczos algorithm. Except for the third application, all computations have been performed in double precision.

4.1 The Double Cross

A collection of validation models for finite element solutions in structural dynamic analyses is given in [29]. The proposed problems correspond to $K\phi = \lambda M\phi$ (free vibration analysis), where K and M are the structural stiffness and mass matrices, $\omega = \sqrt{\lambda}$ is a *free vibration frequency* and ϕ is the corresponding *mode shape*. The in plane pin-ended double cross shown in Figure 8 is one of those test cases. Due to the characteristics of the model, there are many repeated and close eigenvalues. For example, the third frequency has multiplicity four. In [29] the discretization of each arm of the double cross was performed with 4 beam elements, resulting in a system with 83 equations. In the present work the discretization has been performed with 40 equal elements per arm, leading to a problem with dimension 947. The resulting matrix K has 13160 entries and M is diagonal. With a finer mesh, frequencies that can not be represented by the coarse mesh are then introduced. The Figure 9 shows the eigenvalue distribution computed with Matlab for this problem: there are 56 eigenvalues in the range $[10^3, 10^7]$, while the remaining are greater than 10^7.

Table 2 shows the results of some experiments performed with the double cross on a Sun Sparc 10/41. We have measured the CPU time, which is given in seconds, the number of steps performed and the number of solutions converged, for different combinations of m and p. The number of steps and solutions are given between parentheses. The eigenvalue interval was set to $[0, 0]$ and spectral slicing was not used. Random entries were assigned to the starting vectors. Each CPU time listed includes the factorization $K = A_\sigma = LDL^T$, which required 0.6 s, with K stored in a skyline way. In theory, the basic Lanczos algorithm can not detect eigenvalue multiplicities [33]. However, solutions agreeing in many digits are generally achieved due to roundoff errors. It is thus interesting to note in Table 2 that, in spite of the repeated solutions, the best overall performance was obtained with $p = 1$. All the same, useful information can be extracted from those experiments. To begin with, the CPU time variation is smoother for larger block sizes. It is explained by the fact that solutions converged with larger basis sizes when $p > 2$, therefore with an initial overhead. Another thing is that the reduced problem is easier to solve when $p = 1$, which makes a difference when n is relatively small. Finally, the costs for the basis orthogonality control played an important role. Let us consider j blocks of vectors for a modified partial reorthogonalization. Then, $j \times p \times p$ inner products and vectors updatings would be required for the reorthogonalization of Q_{j+1}. Therefore, for bases of dimension $j \times p$, the costs of reorthogonalizing are proportional to p.

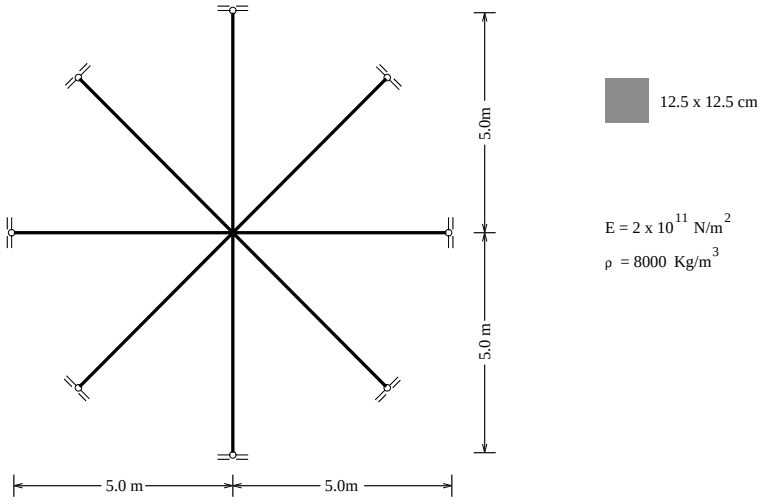

Figure 8: The double cross.

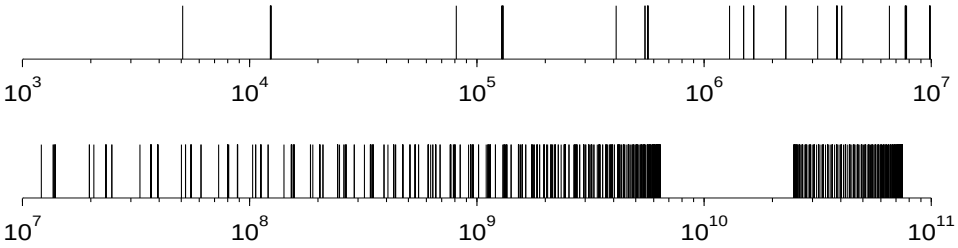

Figure 9: Eigenvalue spectrum for the double cross (logarithmic scale).

Table 2: Computational performance for the double cross.

m	p					
	1	2	3	4	5	6
5	1.6	1.4	3.8	4.2	4.0	5.4
	(22-8)	(9-5)	(14-14)	(11-11)	(8-8)	(8-10)
10	2.5	2.2	3.8	4.2	6.3	5.3
	(33-12)	(14-12)	(14-14)	(11-11)	(11-17)	(8-10)
15	2.7	4.1	5.5	5.1	6.5	8.1
	(34-22)	(22-20)	(17-15)	(12-15)	(11-17)	(10-16)
20	2.8	4.2	6.8	7.4	8.9	11.0
	(34-22)	(22-20)	(19-21)	(15-24)	(13-24)	(12-24)
25	4.9	6.0	9.9	12.0	11.0	13.0
	(50-27)	(27-30)	(23-30)	(19-31)	(14-26)	(13-26)
30	6.2	6.0	9.9	12.0	14.0	16.0
	(55-55)	(27-30)	(23-30)	(19-31)	(16-34)	(14-34)

4.2 A SVD estimation

In this application we deal with a standard eigenproblem. In particular, BLZPACK is employed to estimate some singular values and vectors of the matrix WM3, which is available in the Harwell-Boeing Sparse Matrix Collection [12]. The singular value decomposition is a clever way of determining the matrix rank and it is therefore a useful tool in the analysis of systems of equations [15]. WM3 is related with a model of the world economy and has dimension 207×260. Its pattern is given in Figure 10, which contains 2948 entries, and its singular values distribution computed with Matlab is given in Figure 11.

For a real $r \times s$ matrix C there are two possible computational strategies for obtaining the singular value decomposition. The first is based on the product C^TC, while the second requires the $(r+s) \times (r+s)$ operator defined as

$$A = \begin{bmatrix} 0 & C \\ C^T & 0 \end{bmatrix}.$$

The latter approach presents better numerical properties than the former [15], with the associated problem $Ax = \lambda x$ corresponding to

$$\begin{bmatrix} 0 & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} x^{(1)} \\ x^{(2)} \end{Bmatrix} = \begin{Bmatrix} Cx^{(2)} \\ C^Tx^{(1)} \end{Bmatrix} = \lambda \begin{Bmatrix} x^{(1)} \\ x^{(2)} \end{Bmatrix},$$

whose eigenvalues appear as pairs $(-\lambda, +\lambda)$, where $|\lambda|$ is a singular value of C and $x^{(1)}$ and $x^{(2)}$ are the corresponding right and left singular vectors [9, 15]. Moreover, thinking in terms of a Krylov subspace, the matrix A does not need to be explicitly formed. One can keep C and perform computations as follows

$$Aq_j = A \begin{Bmatrix} q_j^{(1)} \\ q_j^{(2)} \end{Bmatrix} = \begin{Bmatrix} Cq_j^{(2)} \\ C^Tq_j^{(1)} \end{Bmatrix},$$

where $q_j^{(1)}$ and $q_j^{(2)}$ have lengths r and s, respectively, thus generating the Krylov subspace

$$\mathcal{K}(\begin{bmatrix} 0 & C \\ C^T & 0 \end{bmatrix}, \begin{Bmatrix} q_1^{(1)} \\ q_1^{(2)} \end{Bmatrix}, j) = span(\begin{Bmatrix} q_1^{(1)} \\ q_1^{(2)} \end{Bmatrix}, \begin{Bmatrix} Cq_1^{(2)} \\ C^Tq_1^{(1)} \end{Bmatrix}, \begin{Bmatrix} CC^Tq_1^{(1)} \\ C^TCq_1^{(2)} \end{Bmatrix}, \begin{Bmatrix} CC^TCq_1^{(2)} \\ C^TCC^Tq_1^{(1)} \end{Bmatrix}, ...).$$

On the other hand, Cullum *et al.* [9] showed that for the computation of singular and singular vectors with a Lanczos based code, it is appropriate to set $q_1^{(1)} = 0$, if $r > s$, or $q_1^{(2)} = 0$, if $r < s$. Only a multiplication by either C or C^T is thus required at each step (and the projection matrix has zero diagonal entries). Setting the block size to 3 and the starting block as

$$R_0 = \begin{bmatrix} r_0^{(1)} & r_0^{(2)} & r_0^{(3)} \\ 0 & 0 & 0 \end{bmatrix} = Q_1 \mathsf{B}_1$$

with random entries for $r_0^{(1)}$, $r_0^{(2)}$ and $r_0^{(3)}$, the Table 3 lists a set of eigenvalues and residuals for a basis of size 60 (20 steps), computed on a Sun 10/41 workstation. We can verify that some of the triples $(\lambda, x^{(1)}, x^{(2)})$ are already well approximated. Such a strategy is thus useful in the computation of a partial SVD decomposition. However, one should note that BLZPACK is not tuned for this type of application. The objective here is to show that the structure of A can be handled without difficulty due to the flexibility allowed by the reverse communication strategy.

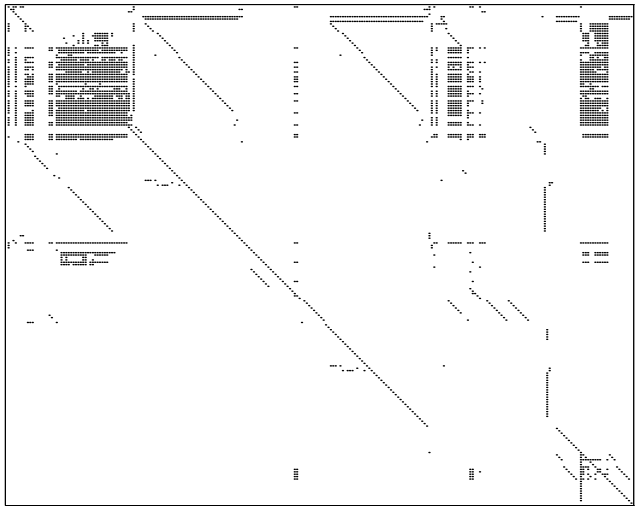

Figure 10: The pattern of WM3.

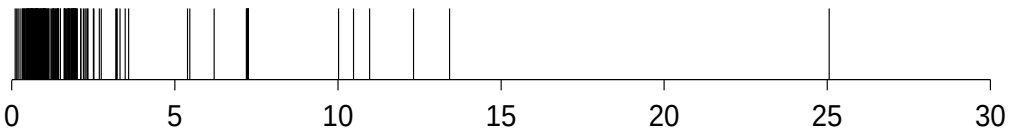

Figure 11: Singular values distribution for WM3.

Table 3: Singular value approximations for WM3.

i	*value*	*residual*
1	2.5058E+01	2.5603E-14
2	1.3422E+01	3.9291E-09
3	1.2318E+01	2.2718E-08
4	1.0970E+01	9.3881E-08
5	1.0478E+01	1.2261E-06
6	1.0022E+01	1.6983E-06
7	7.2500E+00	6.4634E-03
8	7.2464E+00	1.0593E-02
9	7.2459E+00	1.3570E-02
10	7.2249E+00	3.3899E-02

4.3 Citrate Synthase

This application comes from the study of motions of a protein [26]. It is a standard eigenproblem with A corresponding to the product $M^{-1/2}\nabla^2 E M^{-1/2}$, where E represents the potential energy and M is the matrix of the atomic masses (diagonal). We can certainly identify in A an original generalized eigenproblem formulation, but that is the way the problem is retrieved from the CHARMm 21.3 package. The dimension of the problem is three times the number of atoms, the eigenvectors lead to the *normal modes* and the square root of the eigenvalues to the associated *pulsations*. The idea is to express the atomic motions as sinusoidal contributions of the normal modes. We give here some results related with the analysis of the hinge-bending motions for the protein *citrate synthase*, whose ribbon representation is depicted in Figure 12. The dimension of the problem is 25584 and the number of nonzero entries in the upper triangle of the matrix is equal to 3691020 (including the diagonal).

We were interested in the 10 smallest eigenvalues greater than zero, so that an inverse formulation for A was applied. Since the protein is free in the space, the first 6 eigenvalues have no practical interest because they are associated with free (zero energy) molecular motions. The first 16 eigenvalues of *citrate synthase* are listed in Table 4, as well as the corresponding residual norms $\beta_j^{(k)}$. The first 6 eigenvalues are not exactly zero due to roundoff errors in the minimization of E. Based on previous analyses with smaller proteins, a block size equal to 6 was used, with one translation of origin, equal to -0.001. After 20 steps (120 vectors) all required eigenvalues had converged with 671 s of CPU time on one processor of a CRAY C90, including the factorization of the matrix A, which was stored in a skyline way.

4.4 Block size and spectrum slicing effects

Table 4.4 lists a set of matrices available in the Harwell-Boeing Sparse Matrix Collection [12], which have been employed to examine the effects of the block size and the spectrum slicing strategy previously outlined. Some problems are generalized, which are given by pairs of matrices BCSSTK_ and BCSSTM_ (from structural engineering analyses). In such cases, we have included properties of the matrix B (minimum and maximum entries). Other problems are related with systems of linear equations, for which we have set $B = I$. Information about the eigenvalues obtained for each problem is included. Unless indicated in the table, we have set the number of required solutions arbitrarily and the computation interval to $[0, 0]$, thus seeking for solutions close to 0. We then varied p from 1 to 6 and verified the CPU time spent, which is given in seconds (including factorizations), and the number of runs and factorizations performed, which are given as pairs *a-b*. In some cases $a \neq b$, which means that a factorization was used to validate solutions (using the inertia information) or a restart was performed keeping σ fixed. The routine MA47 available in the Harwell Subroutine Library [1] was used to factor matrices and solve systems of equations. The computations were performed on an IBM Risc 6000/950 workstation.

The experiments have been performed not only with large matrices, but also with matrices presenting some numerical interest. The matrix B in the first problem (the pair BCSSTK04-BCSSTM04), for instance, is positive semidefinite. Therefore, the Lanczos algorithm reaches an invariant subspace with $p > 2$, for the number of solutions specified.

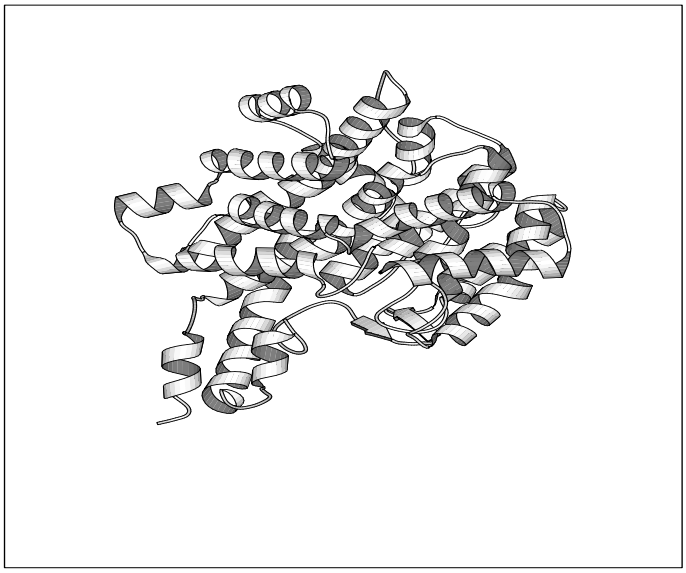

Figure 12: Ribbon representation for citrate synthase.

Table 4: Eigenvalues of citrate synthase.

vector	λ	*residual*
1	-2.4910E-07	7.1047E-17
2	-1.4114E-07	1.5049E-15
3	-5.8901E-08	1.5781E-14
4	-9.7219E-09	1.9154E-14
5	1.3586E-08	6.4485E-15
6	6.1083E-08	2.0790E-15
7	5.6842E-04	9.1855E-15
8	8.4499E-04	1.5364E-13
9	9.1908E-04	4.3734E-13
10	1.0837E-03	3.9170E-12
11	1.2225E-03	1.9194E-11
12	1.3587E-03	3.2015E-11
13	1.4104E-03	3.7157E-10
14	1.5565E-03	9.4498E-10
15	1.7724E-03	6.9597E-10
16	1.8248E-03	3.7490E-09

On the other hand, the first eigenvalue of the matrices BCSSTK14 and BCSSTK16 has a very high multiplicity. The algorithm tries to extend the boundaries of the computational interval and validate the solutions found, detects that there are missing solutions (due to the high multiplicity of the first eigenvalue), and performs a set of translations aiming at identifying possible subintervals containing the missing eigenvalues.

PLAT362 and PLAT1919 are related with oceanic modelling. The first matrix is quasi-singular, but the fact that we have set $\sigma_1 = 0$ did not pose major difficulties. The second matrix is singular and the literature [25] refers to solutions of interest in the middle of the spectrum, namely in the intervals $[0.000025, 0.0001]$ and $[0.0001, 0.24]$. We have then defined these 2 intervals (one at a time) for the algorithm and asked for 100 solutions. In the first case, the actual number of solutions in the interval (64) was less than the number required, in contrast with the second (572 eigenvalues in the interval).

Two problems, BCSSTK25 and NOS6, required special attention. Preliminary analyses performed with the package revealed the existence of very large clusters in the lower end of the spectrum: BCSSTK25 has 132 eigenvalues in the range $[9.6140 \times 10^{-4}, 9.8624 \times 10^{-4}]$; NOS7 has 57 eigenvalues in the range $[1.0000, 1.0003]$. In order to compute those eigenvalues and associated eigenvectors, the limits of the computational interval had to be set very close to the boundaries of the clusters. The imposition of such a constraint is twofold. First, solutions far from the cluster may converge before the eigenvalues in the cluster. By limiting the region of search, the code will discard other solutions. Second, the shift selection will be based only on the information obtained for the cluster, thus avoiding unnecessary factorizations.

5 Conclusions

This work described BLZPACK, an implementation by blocks of the Lanczos algorithm intended for the determination of a set of eigenvalues and associated eigenvectors of real, sparse, symmetric matrices. The package can be applied either to the standard problem $Ax = \lambda x$ or to the generalized problem $Ax = \lambda Bx$. One of the main objectives of the implementation was to provide a neat interface for the user. Therefore, the package supports a set of interaction levels, so the user can set an automatic spectrum slicing, define the threshold for convergence and specify starting vectors, for instance.

Another important point is that the matrices A and B are not required internally in the package. This means that each time a computation involving either A or B has to be performed, the control is returned to the user. Such calculations can be either matrix-vector products (the second application in the previous section) or solutions of systems of linear equations (shift-invert approach). Since the matrices A and B are kept outside the code, the user is free to employ specific storage strategies or experiment with different factorizations routines.

In the applications section, the utilization and performance of the code was examined by means of distinct study cases. In general, for the approach adopted for the shift strategy, and considering the computers on which the experiments were performed, p up to 3 showed to be a good compromise in terms of performance. It should be noted that even for $p = 1$ the package has a block structure and do not profit from particularities this case offers .

Table 5: Block size and spectrum slicing effects for $Ax = \lambda Bx$ (CPU time and number of runs-factorizations)

matrix	n	entries	m	p						characteristics
				1	2	3	4	5	6	
BCSSTK04	132	1890	25	1.1	1.2	1.0	1.0	1.0	1.0	B is diagonal with 66 non zero entries, invariant subspace found for $p > 2$
				2-2	1-2	1-1	1-1	1-1	1-1	$\lambda_1 = 4.3265 \times 10^1$, $\lambda_2 = 4.3850 \times 10^1$, $\lambda_3 = 4.9454 \times 10^1, \ldots \lambda_{66} = 3.3145 \times 10^5$
BCSSTK09	1083	9760	50	15	15	15	16	16	17	B is diagonal positive definite ($b_{min} = 2.5902 \times 10^{-8}$, $b_{max} = 2.5902 \times 10^{-4}$)
				2-2	1-2	1-2	1-2	1-2	1-2	$\lambda_1 = 2.9069 \times 10^7$, $\lambda_2 = \lambda_3 = 1.2070 \times 10^8, \ldots \lambda_{50} = 1.3440 \times 10^{10}$
BCSSTK14	1806	32630	50	32	29	39	34	29	27	$B = I$
				9-8	7-7	6-5	5-3	4-3	3-2	$\lambda_1 = \cdots = \lambda_{40} = 1.0000$, $\lambda_{41} = 3.2865 \times 10^3, \ldots \lambda_{1806} = 1.1923 \times 10^{10}$
BCSSTK15	3948	60882	80	81	92	90	92	100	130	$B = I$
				2-2	2-2	2-2	2-2	2-2	3-2	$\lambda_1 = \cdots = \lambda_6 = 1.0000$, $\lambda_7 = 7.3493 \times 10^2, \ldots \lambda_{80} = 9.5088 \times 10^3$
BCSSTK16	4884	147631	100	260	260	280	270	250	260	$B = I$
				12-14	10-12	8-11	8-10	7-9	7-9	$\lambda_1 = \cdots = \lambda_{74} = 1.0000$, $\lambda_{75} = 1.5895 \times 10^6$
BCSSTK18	11948	80519	100	210	210	220	260	400	410	$B = I$
				3-4	4-4	4-4	4-3	9-7	8-5	$\lambda_1 = 1.2414 \times 10^{-1}$, $\lambda_2 = 1.9817 \times 10^{-1}, \ldots \lambda_{100} = 8.8594 \times 10^{-1}$
BCSSTK21	3600	15100	80	57	63	62	66	73	78	B is diagonal positive definite, ($b_{min} = 6.1819 \times 10^{-6}$, $b_{max} = 1.4661 \times 10^{-4}$)
				2-2	2-2	2-2	2-2	2-2	2-2	$\lambda_1 = 5.1125 \times 10^4$, $\lambda_2 = 1.0974 \times 10^5$, $\lambda_3 = 3.0091 \times 10^5, \ldots \lambda_{80} = 4.0418 \times 10^7$
BCSSTK23	3134	24156	80	150	160	170	170	200	190	B is diagonal positive definite, ($b_{min} = 8.7419 \times 10^{-3}$, $b_{max} = 8.2703 \times 10^6$)
				2-2	2-2	3-2	3-2	2-3	2-3	$\lambda_1 = 44.584$, $\lambda_2 = 72.945$, $\lambda_3 = 94.350, \ldots \lambda_{80} = 6543.9$
BCSSTK24	3562	81736	100	98	150	190	200	150	130	B is diagonal positive definite, ($b_{min} = 5.6170 \times 10^{-8}$, $b_{max} = 1.0153 \times 10^6$)
				2-2	3-2	5-4	6-4	4-3	4-3	$\lambda_1 = 4.3309$, $\lambda_2 = 9.4694$, $\lambda_3 = 11.808, \ldots \lambda_{100} = 352.55$
BCSSTK25	15439	133840	150	490	900	700	940	640	630	B is diagonal positive definite, ($b_{min} = 1.1514 \times 10^{-1}$, $b_{max} = 6.9771 \times 10^8$)
				5-4	8-7	5-5	7-7	5-5	5-5	132 eigenvalues in interval $[9.6140 \times 10^{-4}, 9.8624 \times 10^{-4}]$
NOS3	960	8402	50	10	12	11	11	12	12	$B = I$
				1-2	2-2	2-2	2-2	2-2	2-2	$\lambda_1 = 1.8288 \times 10^{-2}$, $\lambda_2 = 2.4886 \times 10^{-1}$, $\lambda_3 = 2.6718 \times 10^{-1}, \ldots \lambda_{50} = 2.4711 \times 10^1$
NOS6	675	1965	60	17	16	17	24	23	24	$B = I$
				2-3	2-3	2-3	3-4	5-6	6-6	57 eigenvalues in interval $[1.0000, 1.0003]$, $\lambda_{675} = 7.6506 \times 10^6$
NOS7	729	2673	60	16	18	22	19	17	29	$B = I$
				2-2	7-7	5-2	2-3	4-5	10-10	$\lambda_1 = 4.1541 \times 10^{-3}$, $\lambda_2 = 1.0334 \times 10^{-2}, \ldots \lambda_{729} = 9.8640 \times 10^6$
PLAT362	362	3074	30	2.2	2.6	3.0	3.2	3.7	3.8	$B = I$
				2-2	2-2	2-2	2-2	2-2	2-2	$\lambda_1 = \lambda_2 = 3.5548 \times 10^{-12}$, $\lambda_3 = \lambda_4 = 7.2528 \times 10^{-10}, \ldots \lambda_{361} = \lambda_{362} = 7.7428 \times 10^{-1}$
PLAT1919	1919	4831	100	28	26	34	40	48	49	$B = I$
			(a)	2-3	6-5	6-5	9-9	11-9	9-10	$\lambda_1 = 1.44 \times 10^{-16}$, $\lambda_2 = \lambda_3 = 1.09 \times 10^{-13}, \ldots \lambda_{1918} = \lambda_{1919} = 2.9216$
			100	37	36	57	56	63	70	a) 64 eigenvalues in interval $[0.000025, 0.0001]$: $\lambda_{466}, \ldots \lambda_{529}$
			(b)	4-4	5-5	5-4	5-6	11-10	10-9	b) 572 eigenvalues in interval $[0.0001, 0.24]$: $\lambda_{530}, \ldots \lambda_{1101}$

For matrices with $n > 1000$ we have noticed that the solution of the reducing problem involving T_j was negligible compared with other operations, independently of the values assigned to p. On the other hand, the operations required for controlling the level of orthogonality were sometimes important, mainly for problems with big clusters or eigenvalues with high multiplicity. However, when $p \geq 3$ the code automatically sets for level 3 BLAS kernels, which can lead to high performances on many computers. Although we have not performed analyses with huge problems, it is likely that factorizations would dominate the costs, together with solutions of systems of equations.

Acknowledgments

The author would like to thank Yves-Henri Sanejouand, from IRSAMC, *Université Paul-Sabatier*, in Toulouse. The work performed in collaboration with him, related with the study of protein motions, was an important source of inspiration for the development of BLZPACK.

BLZPACK is available at no cost at the URL http://www.nersc.gov/~osni.

References

[1] AEA Technology, Didcot, England. *Harwell Subroutine Library.* Release 11 (July 1993).

[2] W. E. Arnoldi. The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem. *Quarterly of Applied Mathematics*, IX:17–29, 1951.

[3] K.-J. Bathe. *Finite Element Procedures in Engineering Analysis.* Prentice-Hall, Englewood Cliffs, USA, 1982.

[4] M. W. Berry. A Survey of Public-Domain Lanczos-Based Software. In J. D. Brown, M. T. Chu, D. C. Ellison, and R. J. Plemmons, editors, *Proceedings of the Cornelius Lanczos International Centenary Conference*, pages 332–334. SIAM, 1994.

[5] A. Björck. Numerics of Gram-Schmidt Orthogonalization. *Linear Algebra and Its Applications*, 197,198:297–316, 1994.

[6] S. W. Bostic and R. E. Fulton. Implementation of the Lanczos Method for Structural Vibration Analysis on a Parallel Computer. *Computers & Structures*, 25:395–403, 1987.

[7] S.-C. Chang. Lanczos Algorithm with Selective Reorthogonalization for Eigenvalue Extraction in Structural Dynamic and Stability Analysis. *Computers & Structures*, 23:121–128, 1986.

[8] J. Cullum and R. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, volume I Theory, II Programs. Birkhauser, Boston, USA, 1985.

[9] J. Cullum, R. A. Willoughby, and M. Lake. A Lanczos Algorithm for Computing Singular Values and Vectors of Large Matrices. *SIAM J. Sci. Stat. Comput.*, 4:197–215, 1983.

[10] J. W. Daniel, W. B. Cragg, L. Kaufman, and G. H. Stewart. Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization. *Math. of Comp.*, 30:772–795, 1976.

[11] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. Van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM, Philadelphia, USA, 1991.

[12] I. S. Duff, R. G. Grimes, and J. G. Lewis. User's Guide for the Harwell-Boeing Sparse Matrix Collection (Release I). Technical Report TR/PA/92/86, CERFACS, Toulouse, France, 1992.

[13] T. Ericsson and A. Ruhe. The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems. *Mathematics of Computation*, 35:1251–1268, 1980.

[14] G. Gambolati and M. Putti. A Comparison of Lanczos and Optimization Methods in the Partial Solution of Sparse Symmetric Eigenproblems. *Int. J. for Numer. Meth. in Eng.*, 37:605–621, 1994.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, USA, third edition, 1996.

[16] R. G. Grimes, J. G. Lewis, and H. D. Simon. The Implementation of a Block Lanczos Algorithm with Reorthogonalization Methods. Technical Report ETA-TR-91, Boeing Computer Services, Seattle, USA, 1988.

[17] R. G. Grimes, J. G. Lewis, and H. D. Simon. A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Eigenvalue Problems. Technical Report RNR-91-012, Boeing Computer Services, Seattle, USA, 1991.

[18] R. G. Grimes, J. G. Lewis, and H. D. Simon. A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Eigenvalue Problems. *SIAM J. Matrix Anal. Appl.*, 15:228–272, 1994.

[19] V. K. Gupta, J. G. Cole, and W. D. Mock. A Cost-Effective Eigensolution Method for Large Systems with Rockwell Nastran. *Nuclear Engineering and Design*, 78:329–333, 1984.

[20] M. T. Jones and M. L. Patrick. The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem. Technical Report 89-69, ICASE, Hampton, USA, 1989.

[21] M. T. Jones and M. L. Patrick. The Lanczos Algorithm for the Generalized Symmetric Eigenproblem on Shared-Memory Architectures. Technical Report MCS-P182-0990, Argonne National Laboratory, Argonne, USA, 1990.

[22] M. T. Jones and M. L. Patrick. The Use of Lanczos's Method to Solve the Large Generalized Symmetric Eigenvalue Problem in Parallel. Technical Report 90-48, ICASE, Hampton, USA, 1990.

[23] C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *J. of Res. of the Nat. Bur. of Stand.*, 45:255–282, 1950.

[24] R. B. Lehoucq. *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration.* PhD thesis, Rice University, Houston, USA, 1995.

[25] J. G. Lewis. Algorithms for Sparse Matrix Eigenvalue Problems. Technical Report STAN-CS-77-595, Computer Science Dept., Stanford University, Stanford, USA, 1977.

[26] O. A. Marques and Y.-H. Sanejouand. Protein Motions through Eigenanalyses: A Set of Study Cases. Technical Report TR/PA/95/32, CERFACS, Toulouse, France, 1995.

[27] H. G. Matthies. A Subspace Lanczos Method for the Generalized Symmetric Eigenproblem. *Computers & Structures*, 21:319–325, 1985.

[28] R. B. Morgan and D. S. Scott. Preconditioning the Lanczos Algorithm for Sparse Symmetric Eigenvalue Problems. *SIAM J. Sci. Stat. Comput.*, 14:585–593, 1993.

[29] A. Morris. Selected Results from NAFEMS, Dynamics Working Group Free Vibrations, Benchmarks (Part 1). *Benchmark*, pages 12–19, April 1989. Nat. Agency for Finite Element Methods & Standards.

[30] B. Nour-Omid. The Lanczos Algorithm for Solution of Large Generalized Eigenproblem. In T. J. R. Hughes, editor, *The Finite Element Method*, pages 582–630, Englewood Cliffs, USA, 1987. Prentice Hall International Editions.

[31] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen. How to Implement the Spectral Transformation. *Mathematics of Computation*, 48:663–673, 1987.

[32] I. U. Ojalvo. Proper Use of Lanczos Vectors for Large Eigenvalue Problems. *Computers & Structures*, 20:115–120, 1985.

[33] B. N. Parlett. *The Symmetric Eigenvalue Problem.* SIAM (Classics in Applied Mathematics), Philadelphia, USA, 1998.

[34] B. N. Parlett and D. S. Scott. The Lanczos Algorithm with Selective Orthogonalization. *Mathematics of Computation*, 33:217–238, 1979.

[35] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Manchester, England, 1992.

[36] D. S. Scott. The Advantages of Inverted Operators in Rayleigh-Ritz Approximations. *SIAM J. Sci. Stat. Comput.*, 3:68–75, 1982.

[37] H. D. Simon. The Lanczos Algorithm with Partial Reorthogonalization. *Mathematics of Computation*, 42:115–142, 1984.

[38] G. L. G. Sleijpen and H. A. Van der Vorst. A Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems. *SIAM J. Matrix Anal. Appl.*, 17:401–425, 1996.

[39] I. M. Smith and E. E. Heshmati. Use of a Lanczos Algorithm in Dynamic Analysis of Structures. *Earthquake Eng. and Structural Dynamics*, 11:585–588, 1983.

[40] D. C. Sorensen. Implicit Application of Polynomial Filters in a k-step Arnoldi Method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

A User's Guide

The Users' Guide has been moved to a separate file in order to simplify eventual updatings. Please check the directory `doc` in the `blzpack` distribution. In case of trouble, send an e-mail to `osni@nersc.gov`.